International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 2, 2025

Optimized Encryption-Integrated Strategy For Containers Scheduling And Secure Migration In Multi-Cloud Data Centers

T.Rama Krishna, Jakkula Bhanu Prasad, Jinakala Venu, Gunda Uday Kumar

[1]Assistant Professor, Department Of IT, Guru Nanak Institutions Technical Campus (Autonomous), India.

[2,3,4]B.Tech Students, Department Of IT, Guru Nanak Institutions Technical Campus (Autonomous), India

bhanuprasadjakkula@gmail.com

ABSTRACT

Containers are recognized for their lightweight and virtualization efficiency, making them a vital element in modern application orchestration. In this context, the scheduler is crucial in strategically distributing containers across diverse computing nodes. This paper presents a novel two-stage container scheduling solution that addresses node imbalances and efficiently deploys containers. The proposed solution formulates the scheduling process as an optimization problem, integrating various objective functions and constraints to enhance server consolidation and minimize energy consumption. The confidentiality of migrated containers is ensured through encryption, and the associated costs are incorporated into the optimization constraints. This approach ensures security in container scheduling, considering container attributes as input features in our proposed attributes-based encryption model. By carefully selecting containers and destination nodes, this work seeks to establish balance within cloud-based clusters. This contributes to the improvement of container orchestration systems and their effectiveness in real-world scenarios. The proposed solution's efficacy is demonstrated in its ability to efficiently deploy containers in multi-data center cloud environments and seamlessly migrate them between hosts within the same data center or across different data centers. Our results show optimal consolidation with a reduction in the number of running hosts, ranging from 4% to over 18%. Additionally, the solution promotes minimal total power consumption with savings ranging from 3.5 to 16.25 megawatts, while also ensuring balanced server loads.

1-INTRODUCTION

Over the past few years, containers have become popular for simplifying application deployment and management. This virtualization technology provides a consistent runtime environment, enabling seamless application mobility across diverse systems. Additionally, containers are lighter than traditional virtual machines, translating to faster initiation and termination and superior resource utilization.

Container technology is a virtualization method that provides a runtime environment for applications and their dependencies at the process level [1]. The container is composed of two primary layers: the application layer and the base image layer. These layers work in tandem to ensure the smooth operation of the application or microservice [2]. The application layer contains the code for the application itself, and dependencies include codes, binaries, system libraries, and any configuration files needed for the application to run. The base image consists of these files and common system files. The container's ability to run relies on the necessary binaries and libraries in the base image layer. A container engine must be installed on the machine for the container to work. Docker, LNX, OpenVZ, and Containerd are some examples of container engines. Containers can function on bare metal servers or virtual machines, and multiple containers with similar dependencies can be utilizing a common base image. The container platform architecture makes it possible to quickly deploy, migrate, terminate, and replicate containers, boosting the adaptability and scalability of computing systems [3]. Figure 1 illustrates the architecture of the container virtualization platform and its position within a host machine.

Security of containers is important for cloud system providers and end users. There are three levels of data protection for containers: securing the saved container base image (data at rest), securing the running container memory (data in-use), and securing the container data while being migrated (data in-transit). In this research, we specifically focus on ensuring the security of container data while being migrated by encrypting it prior to the migration process at the source host, and decrypting it upon its arrival at the destination.

When migrating between hosts within the same data center or across different ones with varying resource capabilities, it's crucial to have an adaptive encryption algorithm and key size. We are utilizing attribute-based encryption, encompassing attribute-based access control (ABAC) and attribute-based encryption (ABE). In our proposed approach, a customized ABE method has been effective for our needs, as we rely on the cloud management portal to handle authorization control between cloud data centers. To clarify, we cannot consider a static portal configuration for the container migration data block using one encryption type and key size due to the diversity in data center hardware and running container characteristics with user SLA requirements. This highlights the need for an on-the-fly system to set a key size and encryption method type as needed.

Attribute-based access control (ABAC) is an access control approach that authorizes users based on their specific attributes. Attribute-Based Encryption (ABE) is an encryption technique that allows data owners to set access control policies for encrypted data. This method ensures that only individuals with the necessary attributes can decrypt the protected data. As a result, ABE is considered a promising access control mechanism [5]. There are two types of Attribute-Based Encryption (ABE): Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) [6]. KP-ABE limits access to encrypted data to a chosen few users with keys. It uses policies in users' private keys. On the other hand, CP-ABE is a type of public-key encryption that defines a user's private key using attributes. The ciphertext is tied to an access policy, and decryption is only possible if the policy and user attributes match. This access control is based on user attributes, which can be issued by attribute authorities in one or multiple domains [7]. CP-ABE schemes are advantageous as they can cover a large number of users. Multi-authority CP-ABE schemes are well-suited for applications like e-health and e-government where users have attributes from multiple domain authorities.

2-LITERATURE SURVEY

Title: Container-based virtualization for real-time industrial systems-A systematic review
Year: 2023
Author: R. Queiroz, T. Cruz, J. Mendes, P. Sousa and P. Simoes.
Description:
Industrial Automation and Control systems have matured into a stable infrastructure model that has been kept fundamentally unchanged, using discrete embedded systems (such as Programmable Logic Controllers) to implement the first line of sensorization, actuation, and process control and stations and servers providing monitoring, supervision, logging/database and data-sharing capabilities, among others. More recently, with the emergence of the Industry 4.0 paradigm and the need for more flexibility, there has been a steady trend towards virtualizing some of the automation station/server components, first by using virtual machines and, more recently, by using container technology. This trend is pushing for better support for real-time requirements on enabling virtualization technologies such as virtual machines and containers.
This article provides a systematic review on the use of container virtualization in real-time environments such as cyber-physical systems, assessing how existing and emerging technologies can fulfill the associated requirements. Starting by reviewing fundamental concepts related to container technology and real-time requirements, it goes on to present the methodology and results of a systematic study of 37 selected papers covering aspects related to the enforcement of real-time constrains within container hosts and the expected task latency on such environments, as well as an overview of container platforms and orchestration mechanisms for RT systems.

Title: Container placement and migration in edge computing: Concept and scheduling models
Year: 2021
Author: O. Oleghe
Description:
Containers are a form of software virtualization, rapidly becoming the de facto way of providing edge-computing services. Research on container-based edge computing is plentiful, and this has been buoyed by the increasing demand for single digit, milliseconds latency computations. A container scheduler is part of the architecture that is used to manage and orchestrate multiple container-based applications on heterogenous computing nodes. The scheduler decides how incoming computing requests are allocated to containers, which edge nodes the containers are placed on, and where already deployed containers are migrated to. This paper aims to clarify the concept of container placement and migration in edge servers and the scheduling models that have been developed for this purpose. The study illuminates the frameworks and algorithms upon which the scheduling models are built. To convert the problem to one that can be solved using an algorithm, the container placement problem in mostly abstracted using multi-objective optimization models or graph network models. The scheduling algorithms are predominantly heuristic-based algorithms, which are able to arrive at sub-optimal solutions very quickly. There is paucity of container scheduling models that consider distributed edge computing tasks. Research in decentralized scheduling systems is gaining momentum and the future outlook is in scheduling containers for mobile edge nodes.

Title: A survey of state-of-the-art multi-authority attribute based encryption schemes in cloud environment.
Year: 2023
Author: R. Gupta, P. Kanungo and N. Dagdee.
Description:
Cloud computing offers a platform that is both adaptable and scalable, making it ideal for outsourcing data for sharing. Various organizations outsource their data on cloud storage servers for availing management and sharing services. When the organizations outsource the data, they lose direct control on the data. This raises the privacy and security concerns. Cryptographic encryption methods can secure the data from the intruders as well as cloud service providers. Data owners may also specify access control policies such that only

the users, who satisfy the policies, can access the data. Attribute based access control techniques are more suitable for the cloud environment as they cover large number of users coming from various domains. Multi-authority attribute-based encryption (MA-ABE) technique is one of the propitious attribute based access control technique, which allows data owner to enforce access policies on encrypted data. The main aim of this paper is to comprehensively survey various state-of-the-art MA-ABE schemes to explore different features such as attribute and key management techniques, access policy structure and its expressiveness, revocation of access rights, policy updating techniques, privacy preservation techniques, fast decryption and computation outsourcing, proxy re-encryption etc. Moreover, the paper presents feature-wise comparison of all the pertinent schemes in the field. Finally, some research challenges and directions are summarized that need to be addressed in near future.

Title: A systematic literature review of attribute based encryption in health services
Year: 2022
Author: R. Imam, K. Kumar, S. M. Raza, R. Sadaf, F. Anwer, N. Fatima, et al.
Description:
The widespread adoption of cloud technology in the healthcare industry has achieved effective outcomes in sharing health records and sensitive data. In recent years, many organizations have prioritized E-Health as their primary goal for advancement in health services. As a result, Attribute Based Encryption (ABE) has emerged as a reliable model for health information exchange across cloud settings. Eventually, it leads to provide acceptable solutions for challenging scenarios like fine-grained access control. Despite ABE significance and its breadth of applications, no systematic and comprehensive survey exists in the literature that covers every variation of ABE in healthcare, highlighting its past and present status. This paper presents a systematic and comprehensive study of ABE works concerning E-Health as the authors rigorously investigate healthcare-focused ABE frameworks and examine them based on various descriptive criteria, along with categorizing them systematically in 10 distinct domains and sub-domains, ultimately offering observations and potential recommendations. The descriptive research design, significant findings along with the suggested future works will help future research in ABE to secure the existing E-Health data sharing more effectively. The present study will also facilitate researchers and practitioners to comprehend the past trends and current state of ABE architectures in secure health data-sharing scenarios, as well as the prospect of ABE deployment in most recent technological evolutions.

Title: A survey on multi-authority and decentralized attribute-based encryption
Year: 2022
Author: P. S. K. Oberko, V.-H.-K. S. Obeng and H. Xiong
Description:
The introduction of attribute-based encryption (ABE) targets to achieve the implementation of single-to-numerous encryption; however, the sole authority challenge and the issue of distributed management of attributes are bottlenecks to its realization. Multi-authority attribute-based encryption (MA-ABE) where various attribute authorities (which may be independent of each other) control different attribute universe and are involved in the administration of attribute keys for decryption provides the necessary platform to undertake the implementation of fine-grained access regulation over shared data while achieving single-to-numerous encryption. In recent years, research into MA-ABE has seen rapid advancement, and we believe that it is a suitable solution to thwarting the key escrow problem as well as the problem of distributed management of attributes. This paper offers a thorough survey and examines the state-of-the-art of some traditional ABE as well as multi-authority attribute-based encryption schemes over the past decade. Furthermore, the survey gives detailed insights on some essential techniques as well as some classic concretely constructed algorithms. Moreover, we discuss an extension (the different directions) of MA-ABE and its progress since its inception. We also provide design principles of MA-ABE and also show comparisons between existing works on areas as security, performance, and functionality. This paper also discusses several interesting open problems. As far as we can tell, no comparable survey on MA-ABE exists in literature so far.

3-REQUIREMENTS ENGINEERING

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY: This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2, 2025

system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

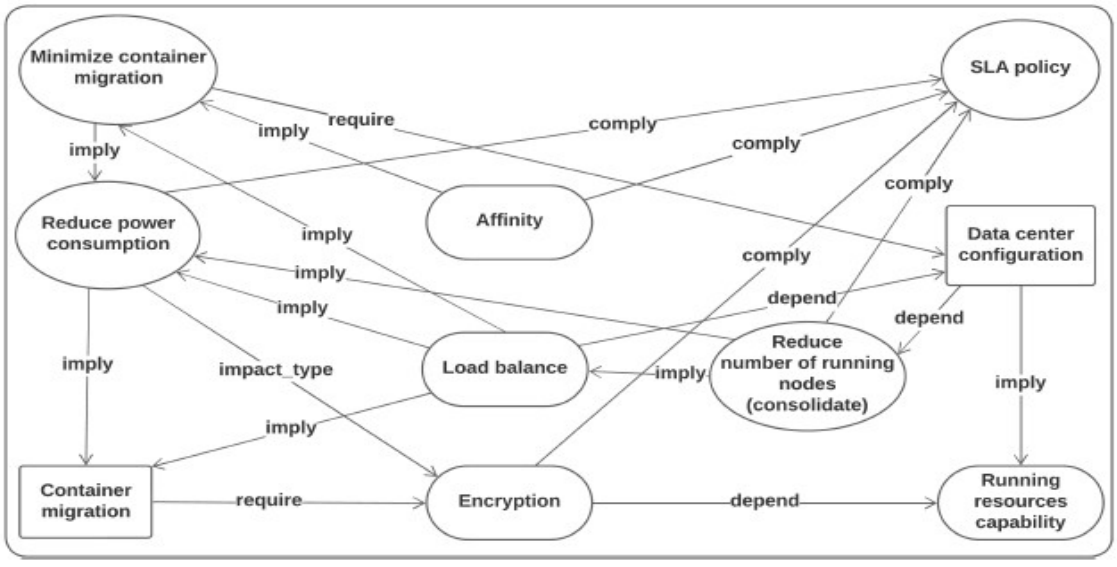
TECHNICAL FEASIBILITY: This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY: The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

These are the requirements for doing the project. Without using these tools & software's we cannot do the project. Therefore, we have two requirements to do the project. They are

- Hardware Requirements.
- Software Requirements.

System Architecture



The optimization model process starts by carefully analyzing a set of predefined SLA Policy requirements. The optimization conditions take into account a variety of factors, including objectives, constraints, application response time, QoS, end-user encryption specification, and supported encryption technologies. By using this information, the optimizer can fulfill all conditions in a selective priority check that matches optimization cost functions and constraints. This process is broken down into two stages. During the first stage (stage 1), the container placement scheduler is formulated to comply with the SLA and optimization goals, which prioritizes fast placement with minimum

HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. Software engineers use them as the starting point for the system design. It shoulds what the system and not how it should be implemented.

SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks, tracking the teams, and tracking the team's progress throughout the development activity.

4-DESIGN ENGINEERING

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

number of running hosts (hybrid first fit and bin packing algorithms). This objective is closely tied to a set of constraints that are addressed in the next stage.

5-DEVELOPMENT TOOLS

the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA, J2EE and J2ME. In this project J2EE is chosen for implementation.

FEATURES OF JAVA

THE JAVA FRAMEWORK

Java is a programming language originally developed by James Gosling at Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

OBJECTIVES OF JAVA

To see places of Java in Action in our daily life, explore java.com.

Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

- Write software on one platform and run it on virtually any other platform
- Create programs to run within a Web browser and Web services
- Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
- Combine applications or services using the Java language to create highly customized applications or services
- Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

Some Ways Software Developers Learn Java

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

ObjectOriented

To be an Object Oriented language, any language must follow at least the four characteristics.

1. Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding addition a feature as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

JAVA SWING OVERVIEW

Abstract Window Toolkit (AWT) is cross-platform

Swing provides many controls and widgets to build user interfaces with. Swing class names typically begin with a J such as JButton, JList, JFrame. This is mainly to differentiate them from their AWT counterparts and in general is one-to-one replacements. Swing is built on the concept of Lightweight components vs AWT and SWT's concept of Heavyweight components. The difference between the two is that the Lightweight components are rendered (drawn) using purely Java code, such as drawLine and drawImage, whereas Heavyweight components use the native operating system to render the components.

Some components in Swing are actually heavyweight components. The top-level classes and any derived from them are heavyweight as they extend the AWT versions. This is needed because at the root of the UI, the parent windows need to be provided by the OS. These top-level classes include JWindow, JFrame, JDialog and JApplet. All Swing components to be rendered to the screen must be able to trace their way to a root window of one of those classes.

EVOLUTION OF COLLECTION FRAMEWORK:

Almost all collections in Java are derived from the **java.util.Collection** interface. Collection

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 2, 2025

defines the basic parts of all collections. The interface states the add() and remove() methods for adding to and removing from a collection respectively. Also required is the toArray() method, which converts the collection into a simple array of all the elements in the collection. Finally, the contains() method checks if a specified element is in the collection. The Collection interface is a sub interface of **java.util.Iterable**, so the iterator() method is also provided. All collections have an iterator that goes through all of the elements in the collection. Additionally, Collection is a generic. Any collection can be written to store any class. For example, Collection<String> can hold strings, and the elements from the collection can be used as strings without any casting required.

There are three main types of collections:

- Lists: always ordered, may contain duplicates and can be handled the same way as usual arrays
- Sets: cannot contain duplicates and provide random access to their elements
- Maps: connect unique keys with values, provide random access to its keys and may host duplicate values

LIST:

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

SET:

Java's java.util.Set interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet, java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a java.util.HashMap to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. java.util.TreeSet uses a red-black tree implemented by a java.util.TreeMap. The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet.

The java.util.Set interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

java.util.SortedSet is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

MAP:

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

THREAD:

Simply put, a *thread* is a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

6-CONCLUSION

This paper introduced a two-stage container scheduling solution that efficiently deploys containers and addresses imbalances across nodes through migration. The solution formulates the scheduling process as an optimization problem and incorporates various objective functions and constraints to improve server consolidation and

reduce energy consumption. Encryption is used to ensure the confidentiality of containers during migration. The cost of encryption and decryption has been included in the optimization constraints of the proposed solution. The solution has been shown to be effective in multi-data center cloud environments, contributing to the improvement of container orchestration systems in real-world scenarios.

REFERENCE

1. "Docker" in What is a Container?—Docker, USA, Sep. 2023, [online] Available: https://www.docker.com/resources/what-container/.
2. R. Queiroz, T. Cruz, J. Mendes, P. Sousa and P. Simões, "Container-based virtualization for real-time industrial systems—A systematic review", ACM Comput. Surv., vol. 56, no. 3, pp. 1-38, Oct. 2023.
3. O. Oleghe, "Container placement and migration in edge computing: Concept and scheduling models", IEEE Access, vol. 9, pp. 68028-68043, 2021.
4. "Weaveworks" in Docker Vs Virtual Machines (VMS): A Practical Guide to Docker Containers and VMS, USA, Sep. 2020, [online] Available: https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms.
5. R. Gupta, P. Kanungo and N. Dagdee, "A survey of state-of-the-art multi-authority attribute based encryption schemes in cloud environment", KSII Trans. Internet Inf. Syst., vol. 17, no. 1, pp. 145-164, 2023.
6. R. Imam, K. Kumar, S. M. Raza, R. Sadaf, F. Anwer, N. Fatima, et al., "A systematic literature review of attribute based encryption in health services", J. King Saud Univ.-Comput. Inf. Sci., vol. 34, no. 9, pp. 6743-6774, Oct. 2022.
7. P. S. K. Oberko, V.-H.-K. S. Obeng and H. Xiong, "A survey on multi-authority and decentralized attribute-based encryption", J. Ambient Intell. Humanized Comput., vol. 13, no. 1, pp. 515-533, Jan. 2022.
8. A. Araldo, A. D. Stefano and A. D. Stefano, "Resource allocation for edge computing with multiple tenant configurations", Proc. 35th Annu. ACM Symp. Appl. Comput., pp. 1190-1199, Mar. 2020.
9. "Kubernetes The Linux Foundation" in Kubernetes: Production-Grade Container Orchestration, USA, Sep. 2023, [online] Available: https://kubernetes.io/.
10. C. Rosen, Docker Swarm vs. Kubernetes: A Comparison. IBM Blog, USA, Jun. 2022, [online] Available: https://www.ibm.com/blog/docker-swarm-vs-kubernetes-a-comparison/.
11. F. Zhang, G. Liu, X. Fu and R. Yahyapour, "A survey on virtual machine migration: Challenges techniques and open issues", IEEE Commun. Surveys Tuts., vol. 20, no. 2, pp. 1206-1243, 2nd Quart. 2018.
12. A. Mirkin, A. Kuznetsov and K. Kolyshkin, "Containers checkpointing and live migration", Proc. Linux Symp., vol. 2, pp. 85-90, 2008.
13. J. Pecholt, M. Huber and S. Wessel, "Live migration of operating system containers in encrypted virtual machines", Proc. Cloud Comput. Secur. Workshop, pp. 125-137, Nov. 2021.
14. F. Brasser, P. Jauernig, F. Pustelnik, A.-R. Sadeghi and E. Stapf, "Trusted container extensions for container-based confidential computing", arXiv:2205.05747, 2022.
15. T. Benjaponpitak, M. Karakate and K. Sripanidkulchai, "Enabling live migration of containerized applications across clouds", Proc. IEEE INFOCOM Conf. Comput. Commun., pp. 2529-2538, Jul. 2020.
16. G. Singh, P. Singh, A. Motii and M. Hedabou, "A secure and lightweight container migration technique in cloud computing", J. King Saud Univ.-Comput. Inf. Sci., vol. 36, no. 1, Jan. 2024.
17. B. Elaine, Recommendation for Key Management Part 1: General, vol. 1, no. 5, pp. 1-171, 2020.
18. D. Patel, B. Patel, J. Vasa and M. Patel, "A comparison of the key size and security level of the ecc and rsa algorithms with a focus on cloud/fog computing", Proc. Int. Conf. Inf. Commun. Technol. Intell. Syst., pp. 43-53, 2023.
19. R. Liu, P. Yang, H. Lv and W. Li, "Multi-objective multi-factorial evolutionary algorithm for container placement", IEEE Trans. Cloud Comput., vol. 11, no. 2, pp. 1430-1445, Jun./Jun. 2023.
20. Y. Mao, W. Yan, Y. Song, Y. Zeng, M. Chen, L. Cheng, et al., "Differentiate quality of experience scheduling for deep learning inferences with Docker containers in the cloud", IEEE Trans. Cloud Comput., vol. 11, no. 2, pp. 1667-1677, Jun./Jun. 2023.
21. L. Deng, Z. Wang, H. Sun, B. Li and X. Yang, "A deep reinforcement learning-based optimization method for long-running applications container deployment", Int. J. Comput. Commun. Control, vol. 18, no. 4, pp. 108-125, 2023.
22. L. Zhu, K. Huang, K. Fu, Y. Hu and Y. Wang, "A priority-aware scheduling framework for heterogeneous workloads in container-based cloud", Applied Intell., vol. 53, no. 12, pp. 15222-15245, Nov. 2022.
23. Y. Han, S. Shen, X. Wang, S. Wang and V. C. M. Leung, "Tailored learning-based scheduling for kubernetes-oriented edge-cloud system", Proc. IEEE INFOCOM Conf. Comput. Commun., pp. 1-10, May 2021.
24. A. Katal, T. Choudhury and S. Dahiya, "Energy optimized container placement for cloud data centers: A meta-heuristic approach", J. Supercomput., vol. 80, no. 1, pp. 98-140, Jan. 2024.

25. A. Bouaouda, K. Afdel and R. Abounacer, "Forecasting the energy consumption of cloud data centers based on container placement with ant colony optimization and bin packing", Proc. 5th Conf. Cloud Internet Things (CIoT), pp. 150-157, Mar. 2022.
26. S. Long, W. Wen, Z. Li, K. Li, R. Yu and J. Zhu, "A global cost-aware container scheduling strategy in cloud data centers", IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 11, pp. 2752-2766, Nov. 2022.
27. Y. Chen, S. He, X. Jin, Z. Wang, F. Wang and L. Chen, "Resource utilization and cost optimization oriented container placement for edge computing in industrial internet", J. Supercomput., vol. 79, no. 4, pp. 3821-3849, Mar. 2023.
28. W. Zhang, L. Chen, J. Luo and J. Liu, "A two-stage container management in the cloud for optimizing the load balancing and migration cost", Future Gener. Comput. Syst., vol. 135, pp. 303-314, Oct. 2022.
29. N. Zhou, F. Dufour, V. Bode, P. Zinterhof, N. J. Hammer and D. Kranzlm\"uller, "Towards confidential computing: A secure cloud architecture for big data analytics and AI", arXiv:2305.17761, 2023.
30. H. Song, J. Li and H. Li, "A cloud secure storage mechanism based on data dispersion and encryption", IEEE Access, vol. 9, pp. 63745-63751, 2021.
31. J. Gabriel, S. Ankermann, M. Seidel and F. H. P. Fitzek, "Transparent storage encryption in kubernetes", Proc. 27th Eur. Wireless Conf., pp. 1-6, Sep. 2022.
32. Q. Deng, X. Tan, J. Yang, C. Zheng, L. Wang and Z. Xu, "A secure container placement strategy using deep reinforcement learning in cloud", Proc. IEEE 25th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD), pp. 1299-1304, May 2022.
33. T. Kong, L. Wang, D. Ma, Z. Xu, Q. Yang and K. Chen, "A secure container deployment strategy by genetic algorithm to defend against co-resident attacks in cloud computing", Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun. IEEE 17th Int. Conf. Smart City IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS), pp. 1825-1832, Aug. 2019.
34. A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", Future Gener. Comput. Syst., vol. 28, no. 5, pp. 755-768, May 2012.
35. J. Luo, Q. Liu, Y. Yang, X. Li, M.-R. Chen and W. Cao, "An artificial bee colony algorithm for multi-objective optimisation", Appl. Soft Comput., vol. 50, pp. 235-251, Jan. 2017.
36. Speed HPC Facility, Jul. 2023, [online] Available: https://github.com/NAG-DevOps/speed-hpc.
37. M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, et al., "Borg: The next generation", Proc. 15th Eur. Conf. Comput. Syst., pp. 1-14, 2020.
38. Borg Cluster Traces From Google, Dec. 2020, [online] Available: https://github.com/google/cluster-data/tree/master.
39. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Softw. Pract. Exper., vol. 41, no. 1, pp. 23-50, Jan. 2011.
40. CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services, Oct. 2023, [online] Available: https://www.cloudbus.org/cloudsim/.
41. SP-SAT Solver, Nov. 2023, [online] Available: https://developers.google.com/optimization/cp/cp_solver.